Life is Tech! スクール

レオ班

チェックシート

2020 春学期

まずはプロジェクトをダウンロードしよう！

GoogleドライブのURL

https://drive.google.com/file/d/1o8YgV0cMwVVwvqtTy4M8TlBsMfOllKI1/view?usp=sharing

ネットが重い場合は直接USBで送るので言ってね！

Mainシーンを開いて実行してみよう！

Unityちゃんが動かない…

今回のチェックワークのテーマ

Unityちゃんを
コントロール
せよ！

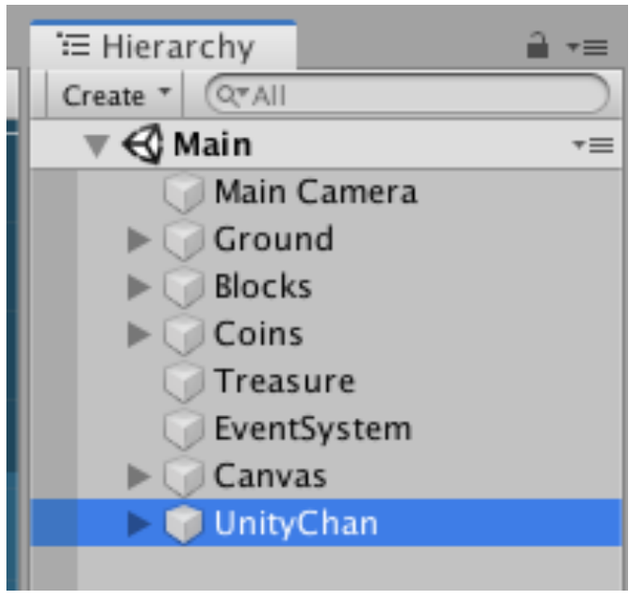
制限時間：30分
合図があるまでこのスライドで待機していてね！

Chapter1

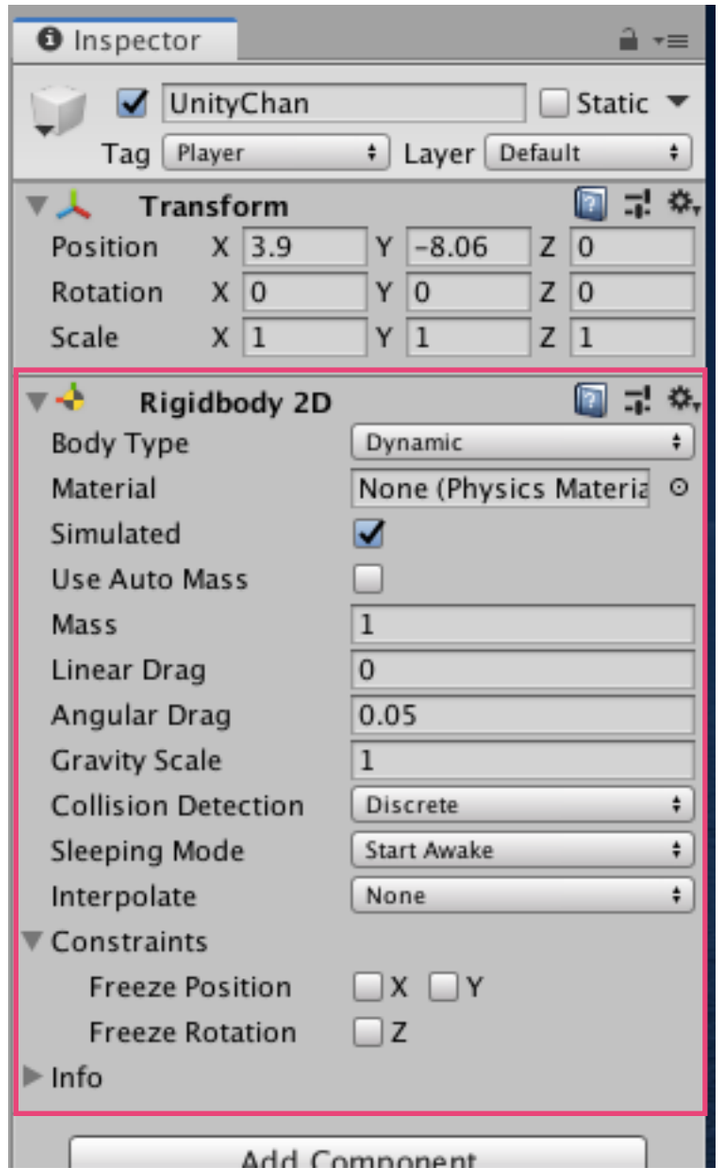
Unityちゃんを
左右キーで移動させよう！

全4問

問題1-0：UnityちゃんにRigidbody2Dをつけよう！



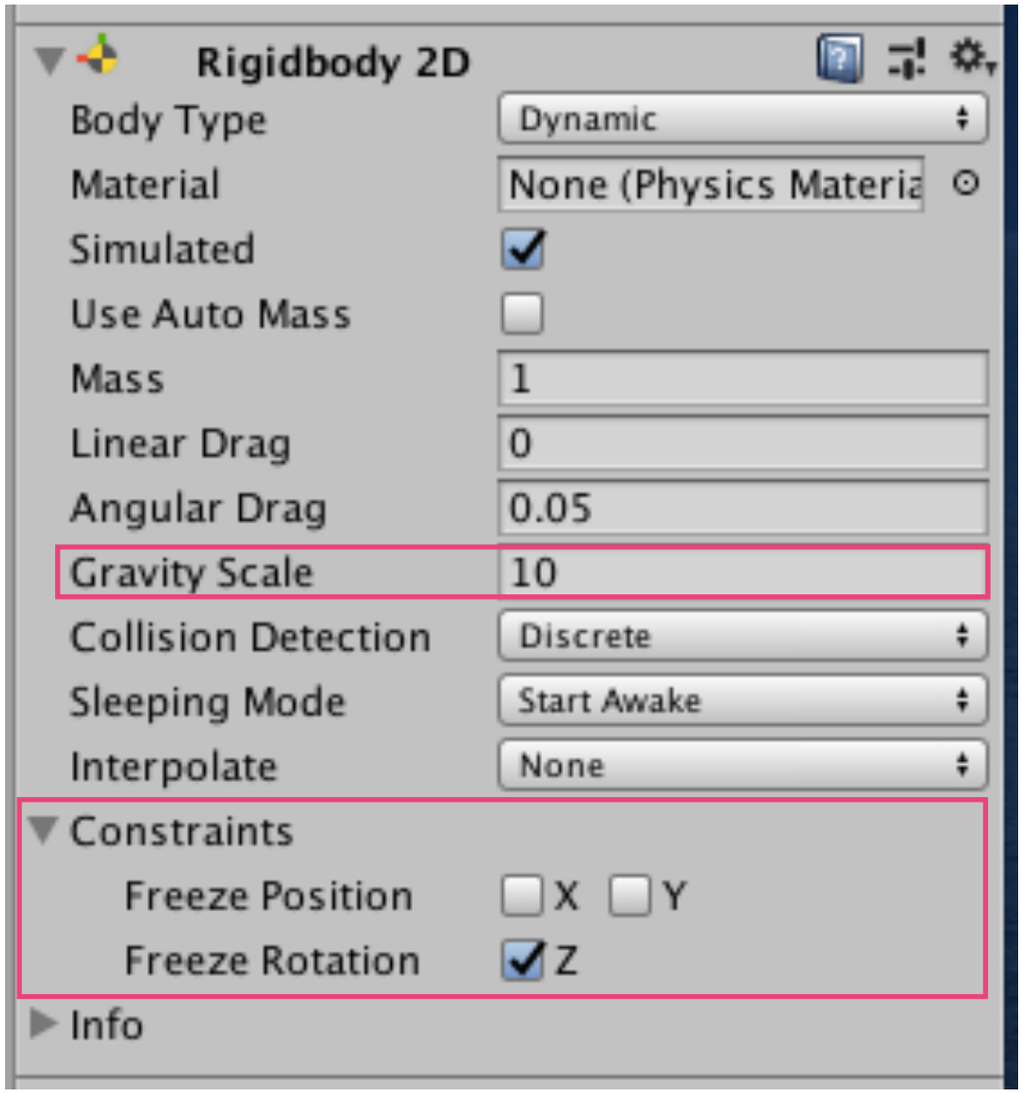
UnityChanを選択



Rigidbody2Dをつける！

□←わからない時は、チェックしてメンターに聞こう！

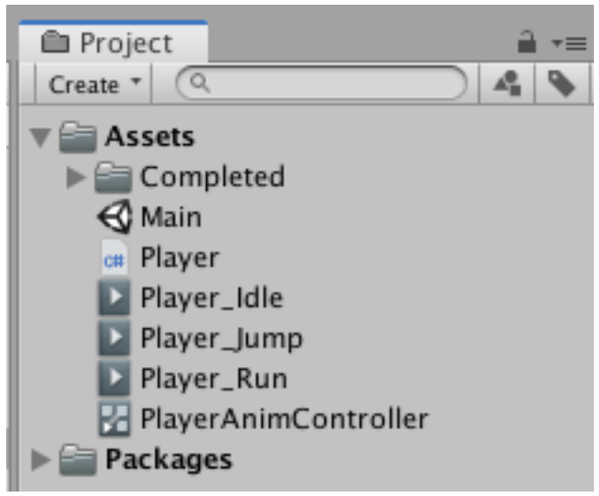
Rigidbody2Dのパラメータを設定しよう！



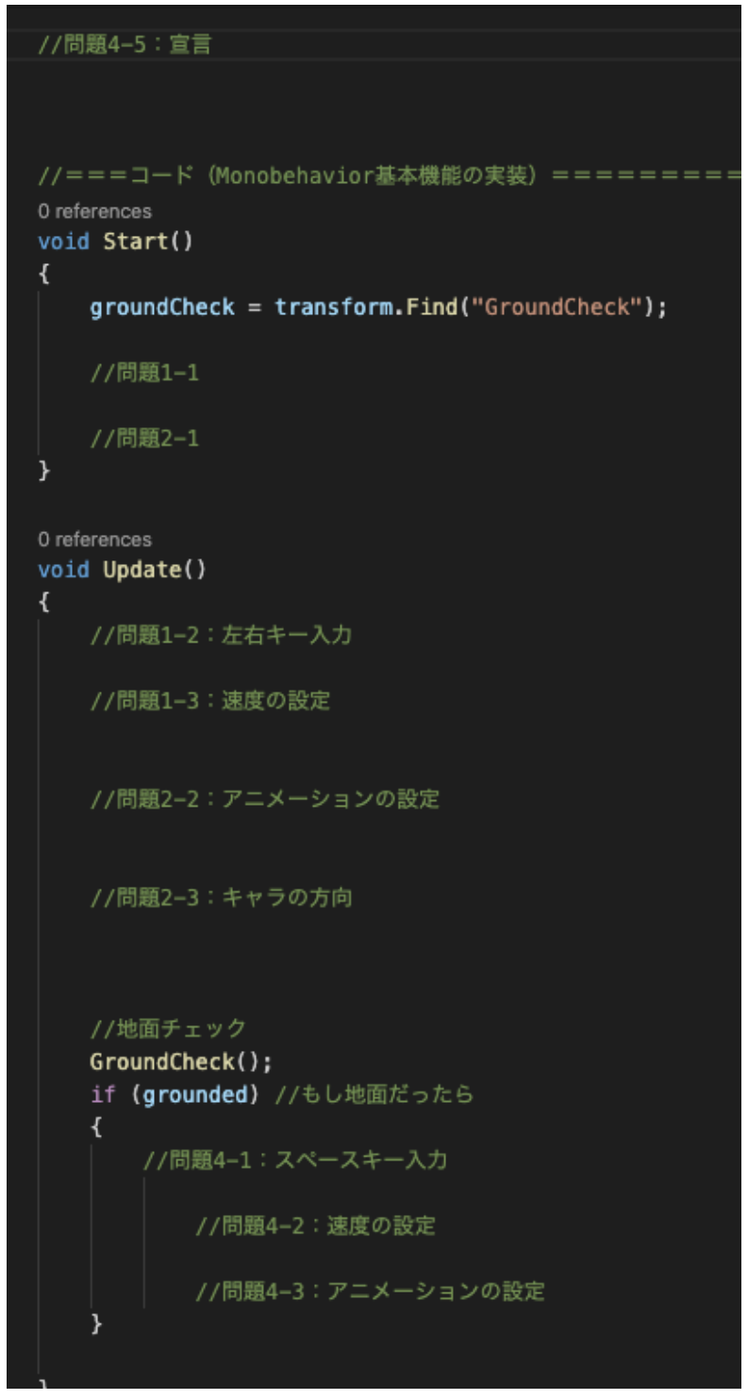
Gravity Scaleを10に！

Constraintsの
Freeze Rotation Z にチェック！

Playerスクリプトを開こう！



Playerスクリプト
を開く



```
//問題4-5：宣言



//===コード（Monobehavior基本機能の実装）==========
0 references
void Start()
{
    groundCheck = transform.Find("GroundCheck");

    //問題1-1

    //問題2-1
}

0 references
void Update()
{
    //問題1-2：左右キー入力

    //問題1-3：速度の設定


    //問題2-2：アニメーションの設定


    //問題2-3：キャラの方向



    //地面チェック
    GroundCheck();
    if (grounded) //もし地面だったら
    {
        //問題4-1：スペースキー入力

            //問題4-2：速度の設定

            //問題4-3：アニメーションの設定
    }
```

問題番号が
書かれているのが
わかるかな？

これから、
対応する問題の場所に
答えを書いていこう！

問題1-1：Rigidbody2DをGetComponentしよう！

・条件

```
Rigidbody2D rb2D; //問題1-1で使う
```

Rigidbody2Dを入れる”箱”は「rb2D」という名前で宣言している

・?の部分に答えを書こう！

```
void Start()
{
    groundCheck = transform.Find("GroundCheck");

    //問題1-1
    ?

    //問題2-1
}
```

□ ←わからない時は、チェックして先に進もう！

注意

次のスライドには答えが書かれています

戻る

もう少し頑張りたい！

進む

できた！ or ギブアップ…

問題1-1：Rigidbody2DをGetComponentしよう！

・条件

```
Rigidbody2D rb2D; //問題1-1で使う
```

Rigidbody2Dを入れる”箱”は「rb2D」という名前で宣言している

・答え

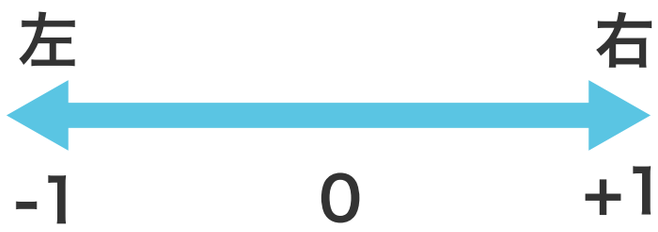
```
void Start()
{
    groundCheck = transform.Find("GroundCheck");

    //問題1-1
    rb2D = GetComponent<Rigidbody2D>();

    //問題2-1
}
```

問題1-2：左右キーの入力を受け取れるようにしよう！

・条件

入力の値は「inputValue」の中に代入する

```
void Update()
{
    //問題1-2：左右キー入力
    float inputValue = Input.GetAxis("Horizontal");
```



左右の入力に応じて
値が -1 から +1 で変化する

□ ←知らなかった時は、チェックして先に進もう！

問題1-3：Rigidbody2Dに速度を与えよう！

・条件

```
float speed = 15.0f; //問題1-3で使う
```

基準となる速度の大きさは「speed」という名前で宣言している

Rigidbody2Dの速度には「入力 × 基準速度」という値を代入する

```
void Update()
{
    //問題1-2：左右キー入力
    float inputValue = Input.GetAxis("Horizontal");

    //問題1-3：速度の設定
    　？　 = new Vector2(　？　, rb2D.velocity.y);
```

□ ←わからない時は、チェックして先に進もう！

注意

次のスライドには答えが書かれています

戻る

もう少し頑張りたい！

進む

できた！ or ギブアップ…

問題1-3：Rigidbody2Dに速度を与えよう！

- 条件

```
float speed = 15.0f; //問題1-3で使う
```

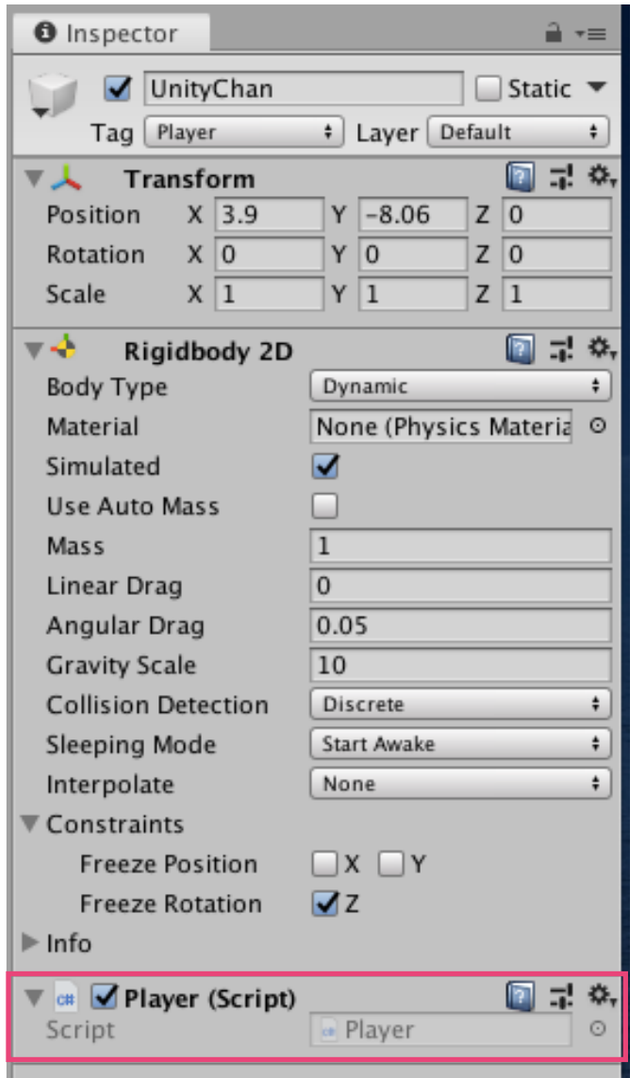
基準となる速度の大きさは「speed」という名前で宣言している

Rigidbody2Dの速度には「入力 × 基準速度」という値を代入する

```
void Update()
{
    //問題1-2：左右キー入力
    float inputValue = Input.GetAxis("Horizontal");

    //問題1-3：速度の設定
    rb2D.velocity = new Vector2(inputValue * speed, rb2D.velocity.y);
```

ちなみに
y方向の速度は「そのまま」

Unityちゃんにスクリプトをつけて実行してみよう！



スクリプトを追加！

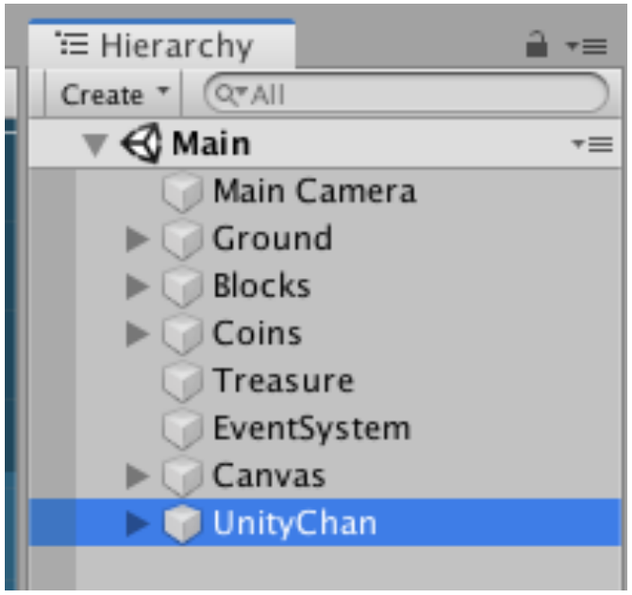
左右キーで移動できるようになったけど、
Unityちゃんが棒立ち…

Chapter2

Unityちゃんに
アニメーションをつけよう！

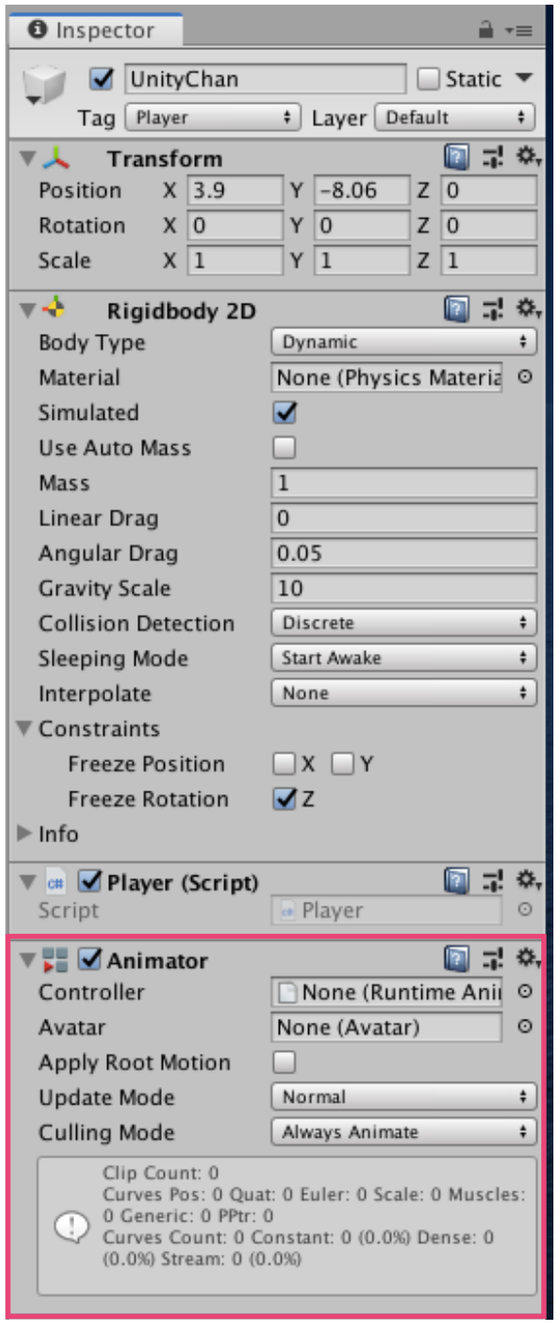
全4問

問題2-0：UnityちゃんにAnimatorをつけよう！



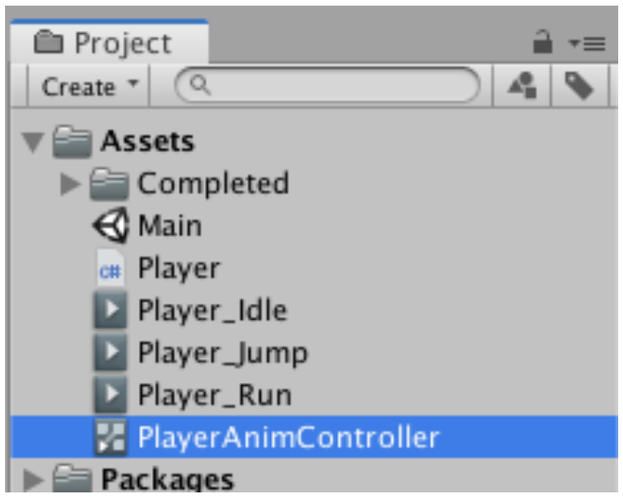
UnityChanを選択

Animatorをつける！

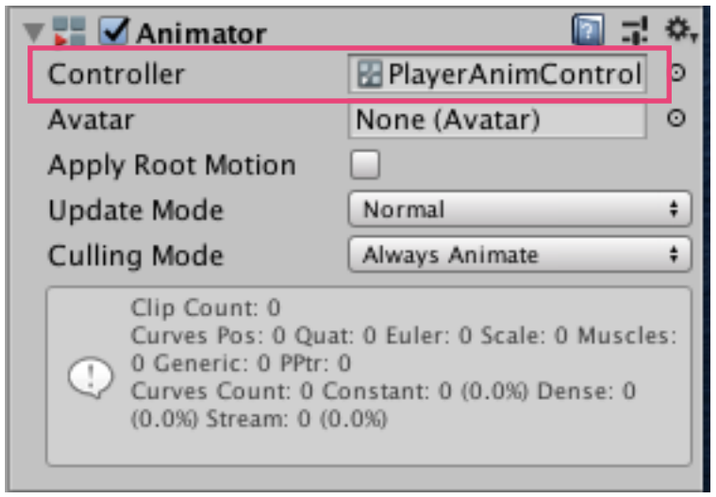


□←わからない時は、チェックしてメンターに聞こう！

Animator Controllerを設定しよう！

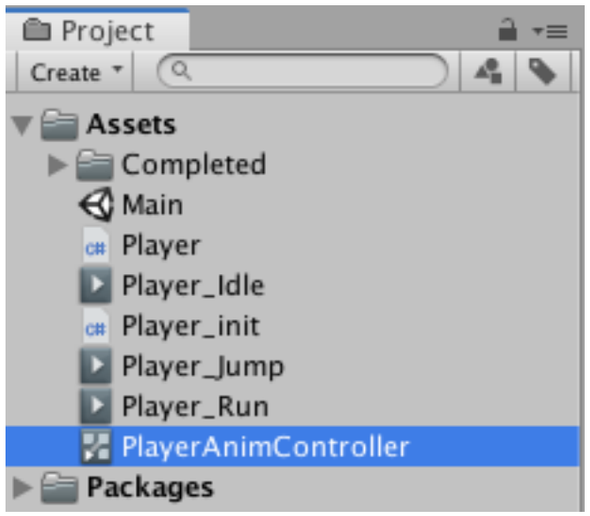


「PlayerAnimController」
という名前で作成している

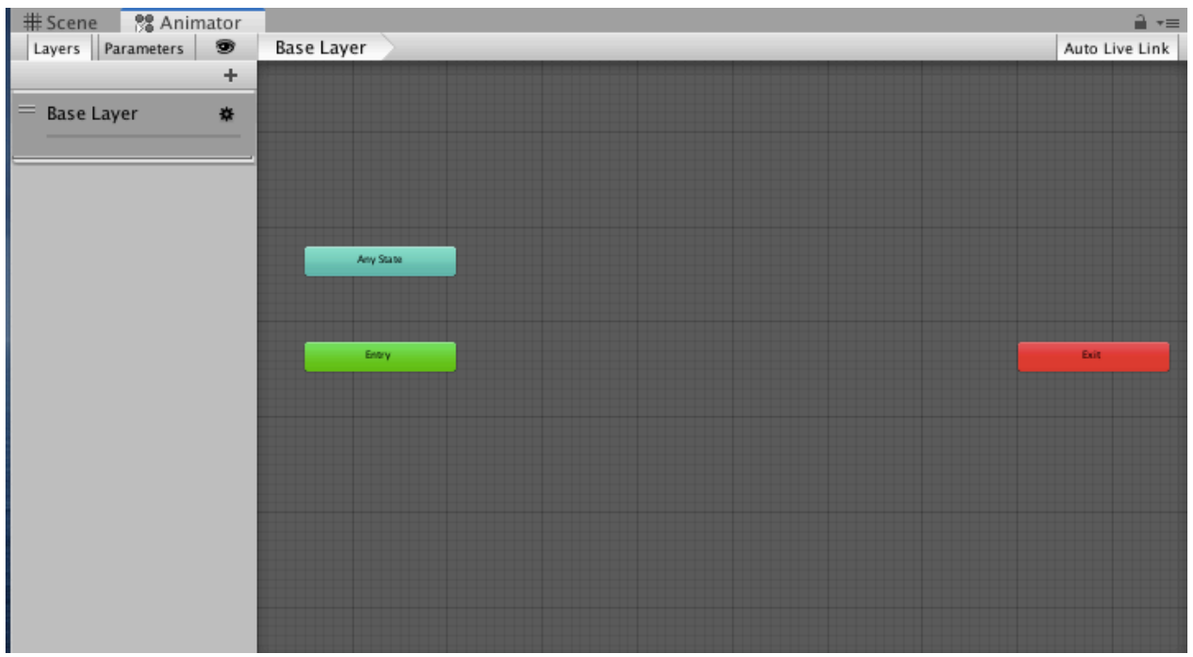


UnityちゃんのAnimatorに
ドラックして設定！

Animator Controllerを設定しよう！

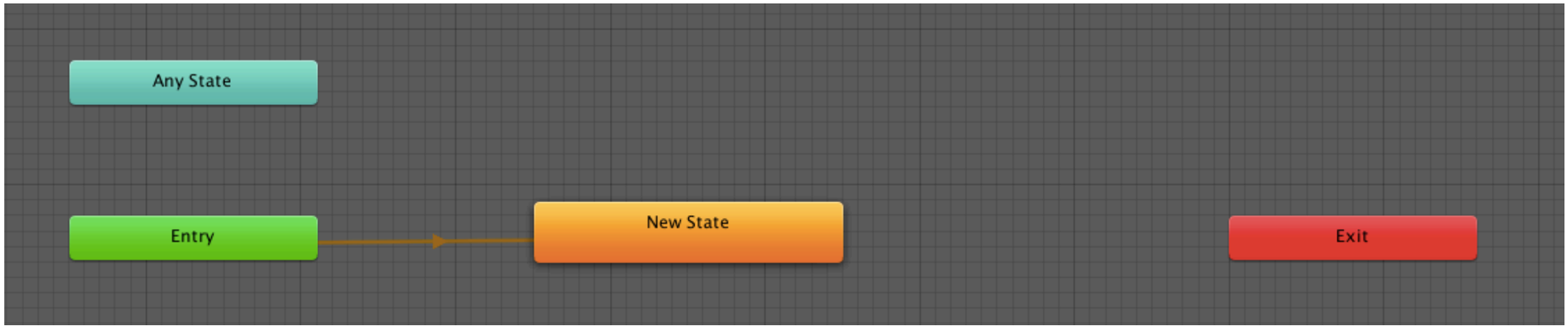


PlayerAnimControllerを
ダブルクリックして開こう！

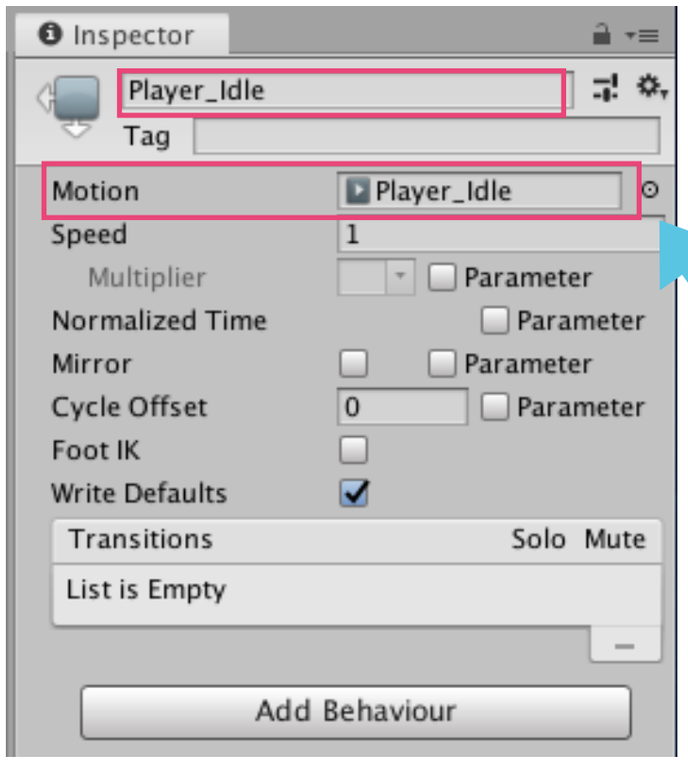


これから中身を
作っていくよ！

Animator Controllerを作ろう！

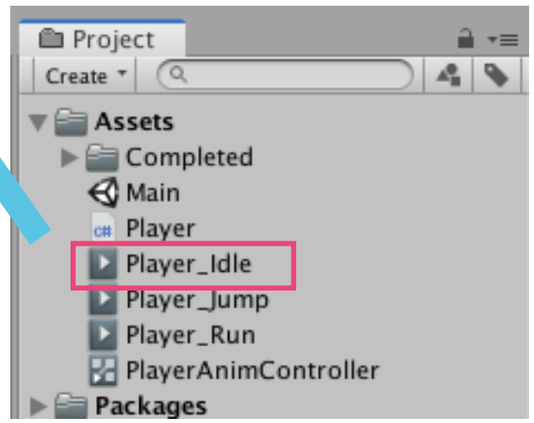


新たにステイトを一つ作ろう！



名前を「Player_Idle」に設定

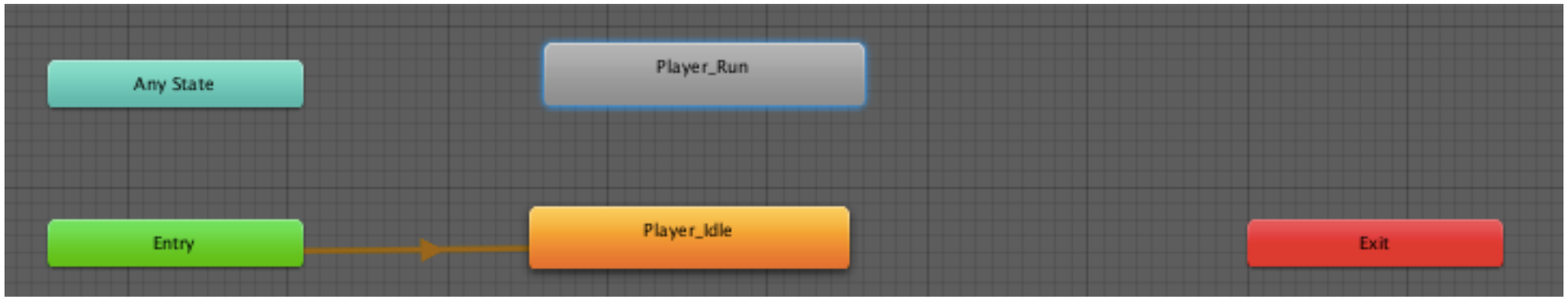
ちなみにIdleとは
「立ち」アニメーションのこと



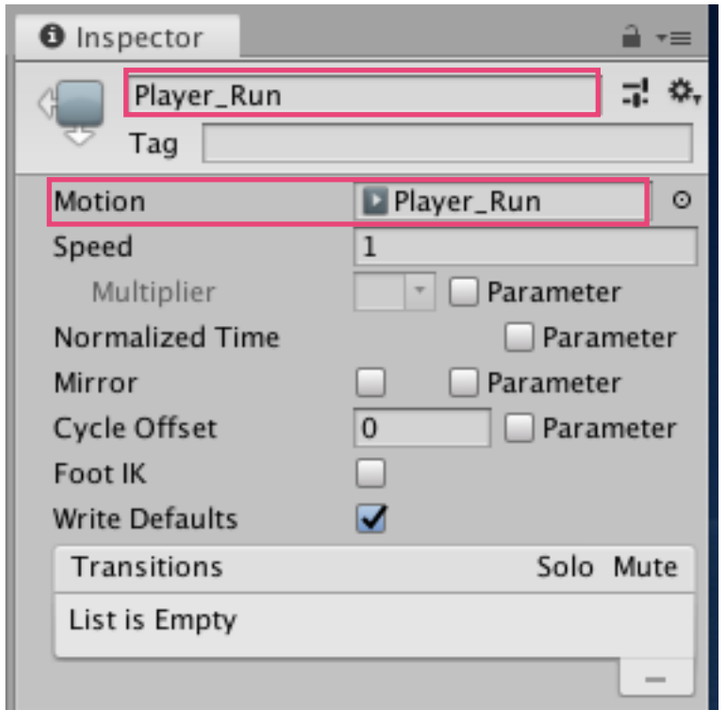
Motionに「Player_Idle」アニメーションを設定！

今回、個々のアニメーションは作成済み

Animator Controllerを作ろう！



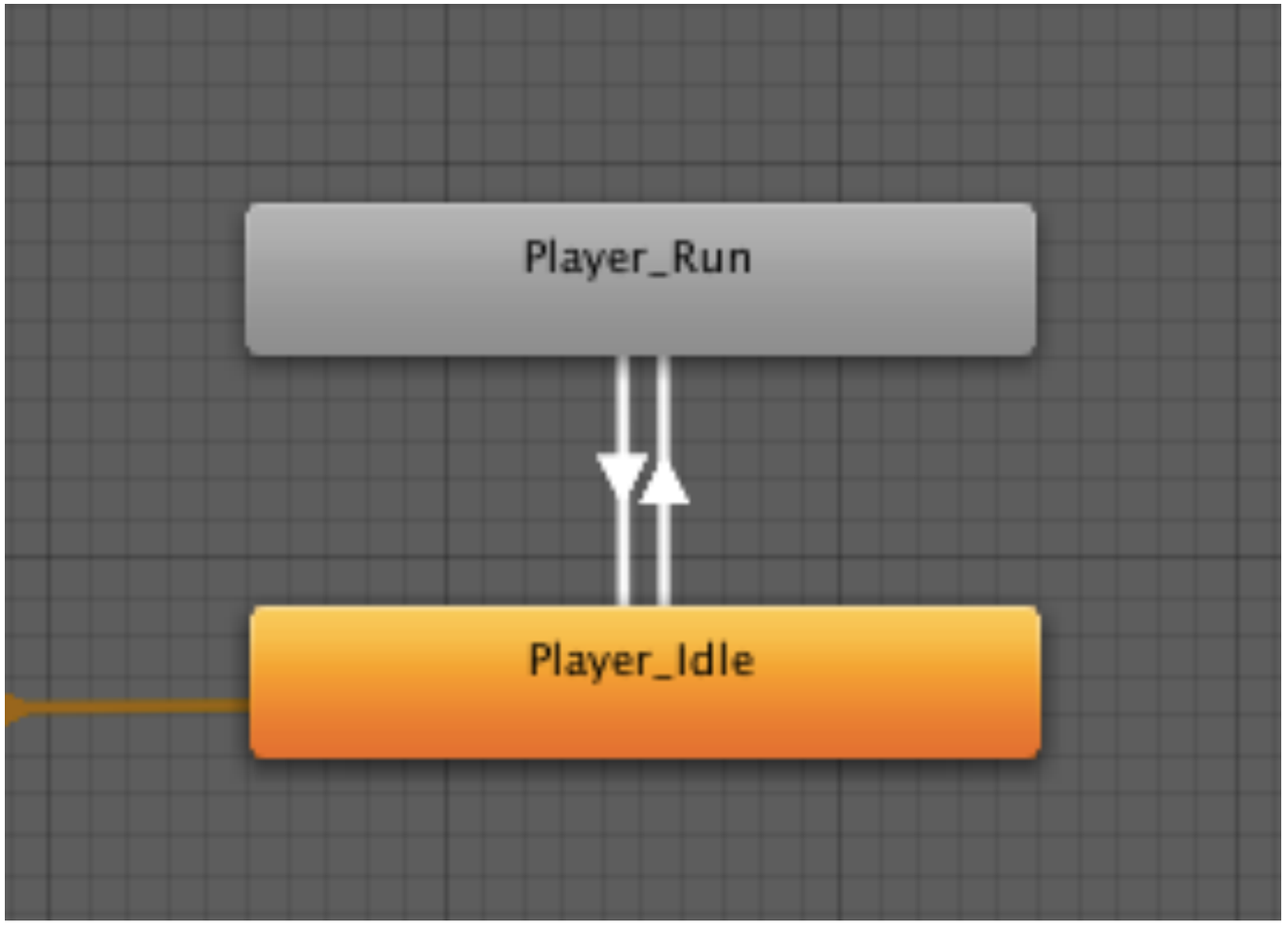
同じ手順でもう1つステイトを作ろう！



名前を「Player_Run」に設定

Motionに「Player_Run」
アニメーションを設定！

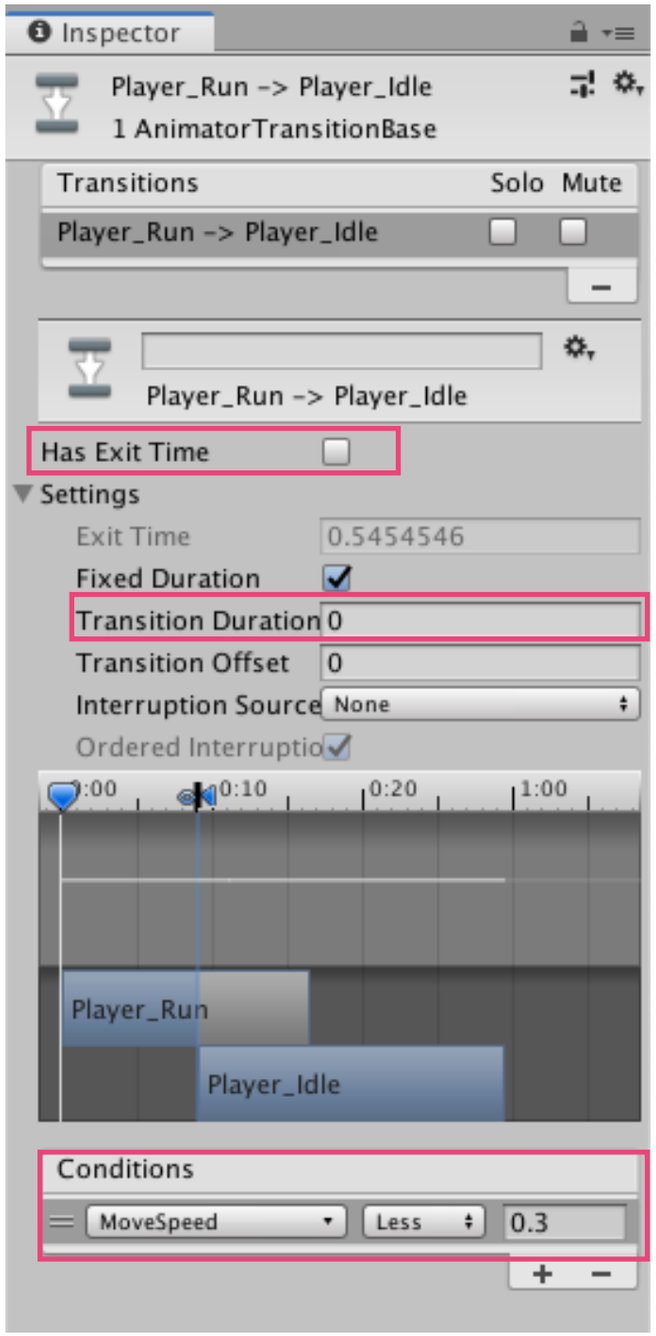
Animator Controllerを作ろう！

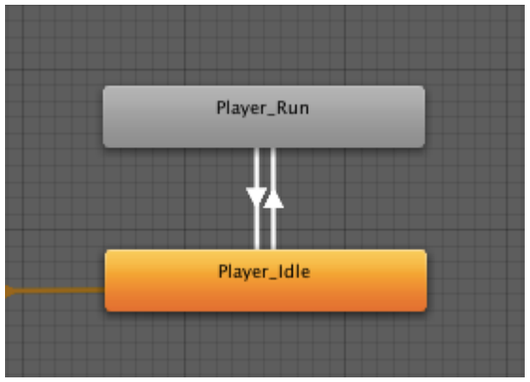


2つのステイトを矢印で繋げよう！

Animator Controllerを作ろう！

Run→Idle

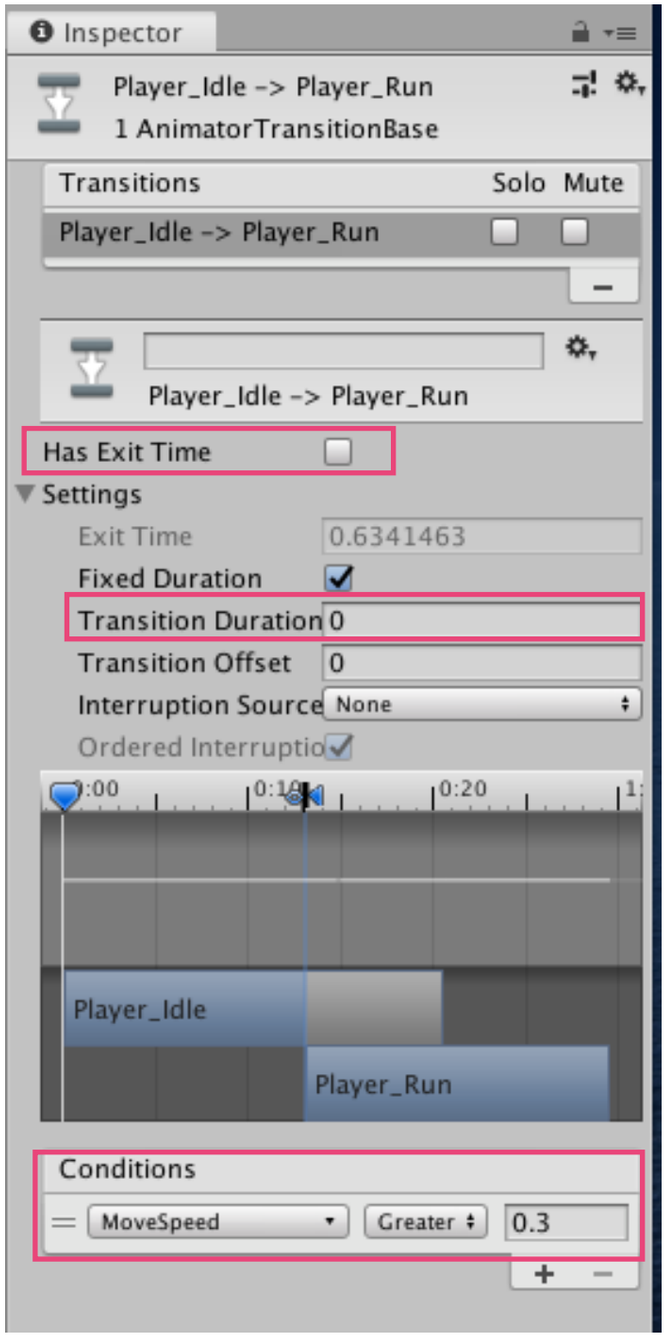




良く見て
パラメータを
設定しよう！

MoveSpeedが0.3を境にして
Idle↔Run
を行き来する設定

Idle→Run



問題2-1：AnimatorをGetComponentしよう！

・条件

```
Animator animator; //問題2-1で使う
```

Animatorを入れる”箱”は「animator」という名前で宣言している

・?の部分に答えを書こう！

```
void Start()
{
    groundCheck = transform.Find("GroundCheck");

    //問題1-1
    rb2D = GetComponent<Rigidbody2D>();

    //問題2-1
    ?
}
```

□ ←わからない時は、チェックして先に進もう！

注意

次のスライドには答えが書かれています

戻る

もう少し頑張りたい！

進む

できた！ or ギブアップ…

問題2-1：AnimatorをGetComponentしよう！

・条件

```
Animator animator; //問題2-1で使う
```

Animatorを入れる”箱”は「animator」という名前で宣言している

・答え

```
void Start()
{
    groundCheck = transform.Find("GroundCheck");

    //問題1-1
    rb2D = GetComponent<Rigidbody2D>();

    //問題2-1
    animator = GetComponent<Animator>();
}
```

Rigidbodyの時と同じ！

問題2-2：Runアニメーションに遷移するようにしよう！

・条件

Animatorでのパラメータの名前は「MoveSpeed」, 型は「Float」
与える値は「inputValue」の絶対値（Mathf.Abs関数を使う）

・?の部分に答えを書こう！

```
void Update()
{
    //問題1-2：左右キー入力
    float inputValue = Input.GetAxis("Horizontal");

    //問題1-3：速度の設定
    rb2D.velocity = new Vector2(inputValue * speed, rb2D.velocity.y);

    //問題2-2：アニメーションの設定
    animator.   ?   (   ?   , Mathf.Abs(   ?   ));
```

□ ←わからない時は、チェックして先に進もう！

注意

次のスライドには答えが書かれています

戻る

もう少し頑張りたい！

進む

できた！ or ギブアップ…

問題2-2：Runアニメーションに遷移するようにしよう！

・条件

Animatorでのパラメータの名前は「MoveSpeed」, 型は「Float」
与える値は「inputValue」の絶対値（Mathf.Abs関数を使う）

・答え

```
void Update()
{
    //問題1-2：左右キー入力
    float inputValue = Input.GetAxis("Horizontal");

    //問題1-3：速度の設定
    rb2D.velocity = new Vector2(inputValue * speed, rb2D.velocity.y);

    //問題2-2：アニメーションの設定
    animator.SetFloat("MoveSpeed", Mathf.Abs(inputValue));
```

「Animator.Set ~ 」でパラメータをセットする！

実行してみよう！

アニメーションは再生されるが、
向きが変わらない…

問題2-3：Unityちゃんの向きが変わるようにしよう！

```
float direction = 1.0f; //問題2-3で使う
```

```
//問題2-3：キャラの方向
if (inputValue != 0.0f)
{
    direction = Mathf.Sign(inputValue);
}
transform.localScale = new Vector3(direction, transform.localScale.y, transform.localScale.z);
```

まずは写そう！

ポイント

- localScaleのxの値が　＋1：右向き / -1：左向き
- 入力「inputValue」の符合（+1 or -1）をlocalScaleに与える

□ ←わからない時は、チェックして先に進もう！

実行してみよう！

向きが変わるようになった！

まだコインには触れない…

Chapter3

Unityちゃんに
当たり判定をつけよう！

全5問

やりたいこと！

コインに当たると、

ここのテキストの文字が変わる！

問題3-1：当たり判定が起きた時に呼ばれる関数を作成しよう！

・条件

コインのColliderでは 「isTrigger」が「オン」になっている

```
//＝＝＝コード（当たり判定）＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝
//問題3-1
0 references
void On      ？      2D(     ？     2D other)
{
    //問題3-2

        //問題3-5


    //問4-4

        //問4-6

}
```

{ }の位置も
よく見て書こう！

□ ←わからない時は、チェックして先に進もう！

注意

次のスライドには答えが書かれています

戻る

もう少し頑張りたい！

進む

できた！ or ギブアップ…

問題3-1：当たり判定が起きた時に呼ばれる関数を作成しよう！

・条件

コインのColliderでは 「isTrigger」が「オン」になっている

```
//＝＝＝コード（当たり判定）＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝
//問題3-1
0 references
void OnTriggerEnter2D(Collider2D other)
{
    //問題3-2

        //問題3-5


//問4-4

        //問4-6
}
```

{ }の位置も
よく見て書こう！

今回は2Dゲームなので名前も「2D」になっているが、3Dと扱いは同じ

問題3-2：コインをタグで判定しよう！

・条件

コインには 「Coin」タグが付けられている

「もし、タグがCoinだったら」というif文を書く

```
//====コード（当たり判定）=====================
//問題3-1
0 references
void OnTriggerEnter2D(Collider2D other)
{
    //問題3-2
    if(          ？          )
    {
        //問題3-5

    }

    //問4-4

        //問4-6

}
```

{ }の位置も
良く見て書こう！

□ ←わからない時は、チェックして先に進もう！

注意

次のスライドには答えが書かれています

戻る

もう少し頑張りたい！

進む

できた！ or ギブアップ…

問題3-2：コインをタグで判定しよう！

・条件

コインには 「Coin」タグが付けられている

「もし、タグがCoinだったら」というif文を書く

```
//====コード（当たり判定）=====================
//問題3-1
0 references
void OnTriggerEnter2D(Collider2D other)
{
    //問題3-2
    if(other.tag == "Coin")
    {
        //問題3-5

    }

    //問4-4

        //問4-6

}
```

{ }の位置も
良く見て書こう！

文字列の「” ”」に注意！

問題3-3：UIを扱えるようにしよう！

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

//問題3-3
using [ ? ];
```

スクリプトの一番上のところに書くよ！

□ ←わからない時は、チェックして先に進もう！

注意

次のスライドには答えが書かれています

戻る

もう少し頑張りたい！

進む

できた！ or ギブアップ…

問題3-3：UIを扱えるようにしよう！

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

//問題3-3
using UnityEngine.UI;
```

スクリプトの一番上のところに書くよ！

問題3-4：変えたいTextをスクリプトと結びつけよう！

・条件

変数の型は 「Text」型

変数名は 「mission1Text」として宣言する

```
public class Player : MonoBehaviour
{
    //===パラメータ=======================
```

~中略~

```
//問題3-4：宣言
0 references
public ？ ？;

//問題4-5：宣言
```

□ ←わからない時は、チェックして先に進もう！

注意

次のスライドには答えが書かれています

戻る

もう少し頑張りたい！

進む

できた！ or ギブアップ…

問題3-4：変えたいTextをスクリプトと結びつけよう！

・条件

変数の型は 「Text」型

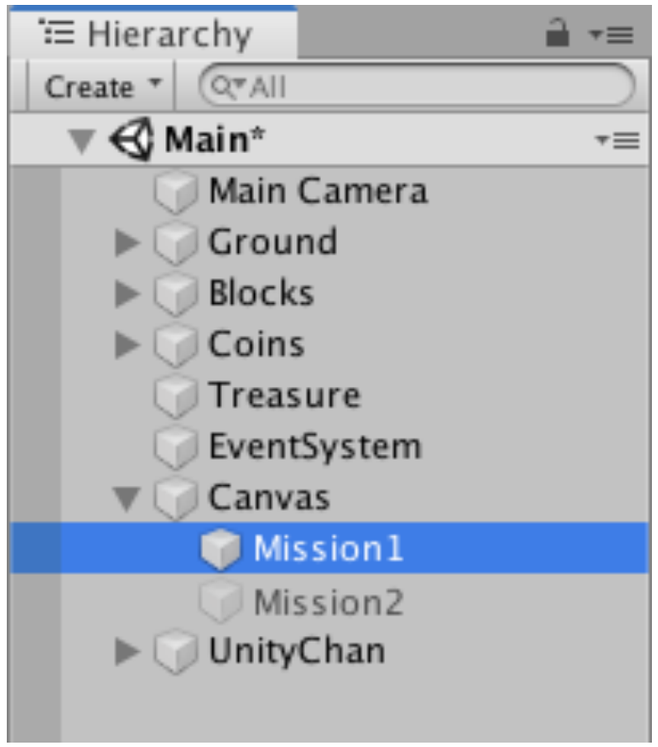
変数名は 「mission1Text」として宣言する

```
public class Player : MonoBehaviour
{
    //===パラメータ=======================
```

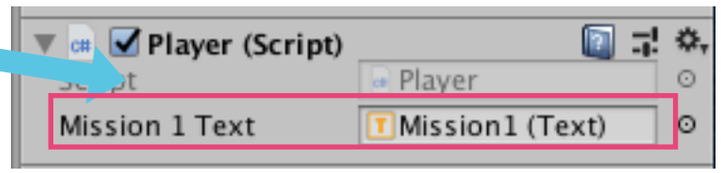
~中略~

```
    //問題3-4：宣言
    0 references
    public Text mission1Text;

    //問題4-5：宣言
```

変えたいTextをスクリプトと結びつけよう！



Mission1テキスト



Playerスクリプトに
関連付けしよう！

問題3-5：Textの中身を変えよう！

・条件

「mission1Text」の中身を「ミッション 1：CLEAR!」に変える

```
//====コード（当たり判定）=====================
//問題3-1
0 references
void OnTriggerEnter2D(Collider2D other)
{
    //問題3-2
    if(other.tag == "Coin")
    {
        //問題3-5
        ？ = "ミッション 1：CLEAR!";

    }

    //問4-4

        //問4-6

}
```

□ ←わからない時は、チェックして先に進もう！

注意

次のスライドには答えが書かれています

戻る

もう少し頑張りたい！

進む

できた！ or ギブアップ…

問題3-5：Textの中身を変えよう！

・条件

「mission1Text」の中身を「ミッション 1：CLEAR!」に変える

```
//====コード（当たり判定）=====================
//問題3-1
0 references
void OnTriggerEnter2D(Collider2D other)
{
    //問題3-2
    if(other.tag == "Coin")
    {
        //問題3-5
        mission1Text.text = "ミッション 1：CLEAR!";

    }

    //問4-4

        //問4-6

}
```

Textコンポーネントのtextパラメータを変更する　　ややこしい…

実行してみよう！

文字が変わる！

コインをゲット！

そして新たなミッションが出現した…

Chapter4

Unityちゃんを
ジャンプさせよう！

全6問

問題4-1：スペースキーの入力を受け取ろう！

・条件

「もしスペースキーが入力されたら」というif文を書く

```
void Update()
{
```

~中略~

```
    //地面チェック
    GroundCheck();
    if (grounded) //もし地面だったら
    {
        //問題4-1：スペースキー入力
        if(          ?          )
        {
            //問題4-2：速度の設定

            //問題4-3：アニメーションの設定
        }
    }

}
```

{ }の位置も
良く見て書こう！

□ ←わからない時は、チェックして先に進もう！

注意

次のスライドには答えが書かれています

戻る

もう少し頑張りたい！

進む

できた！ or ギブアップ…

問題4-1：スペースキーの入力を受け取ろう！

・条件

「もしスペースキーが入力されたら」というif文を書く

```
void Update()
{
```

~中略~

```
    //地面チェック
    GroundCheck();
    if (grounded) //もし地面だったら
    {
        //問題4-1：スペースキー入力
        if(Input.GetKeyDown(KeyCode.Space))
        {
            //問題4-2：速度の設定

            //問題4-3：アニメーションの設定
        }
    }

}
```

{ }の位置も
良く見て書こう！

GetKey”Down”：押した時（1回だけ）

問題4-2：ジャンプの速度を設定しよう！

・条件

Rigidbody2Dに速度を代入する

与える速度は、x方向：そのまま / y方向：+35

```
//地面チェック
GroundCheck();
if (grounded) //もし地面だったら
{
    //問題4-1：スペースキー入力
    if(Input.GetKeyDown(KeyCode.Space))
    {
        //問題4-2：速度の設定
        ？ = new Vector2(？, ？);

        //問題4-3：アニメーションの設定
    }
}
```

□ ←わからない時は、チェックして先に進もう！

注意

次のスライドには答えが書かれています

戻る

もう少し頑張りたい！

進む

できた！ or ギブアップ…

問題4-2：ジャンプの速度を設定しよう！

・条件

Rigidbody2Dに速度を代入する

与える速度は、x方向：そのまま / y方向：+35

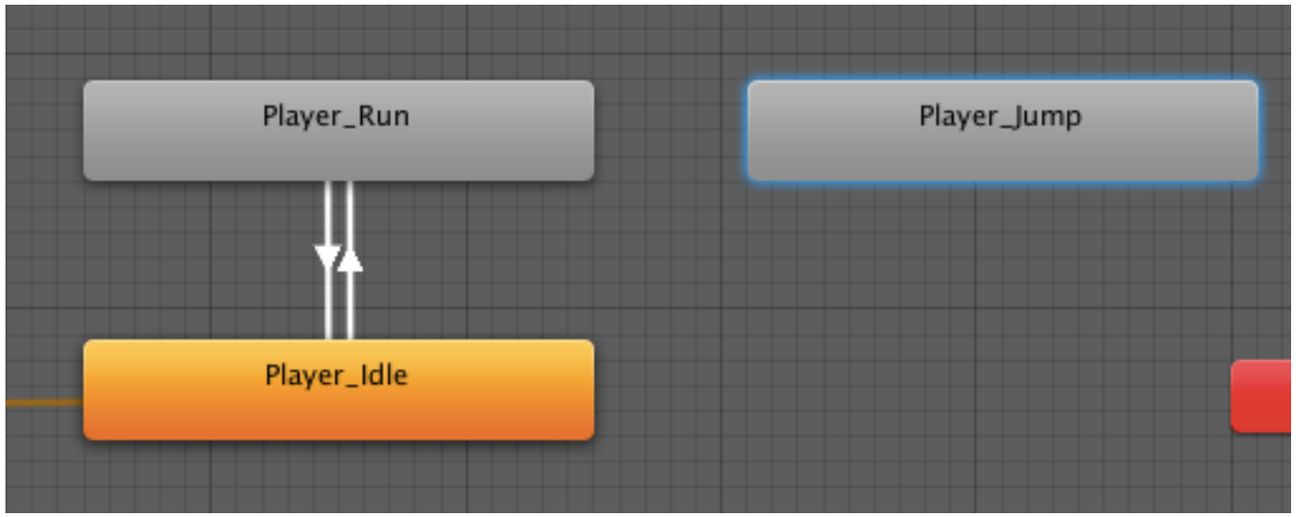
```
//地面チェック
GroundCheck();
if (grounded) //もし地面だったら
{
    //問題4-1：スペースキー入力
    if(Input.GetKeyDown(KeyCode.Space))
    {
        //問題4-2：速度の設定
        rb2D.velocity = new Vector2(rb2D.velocity.x, 35);

        //問題4-3：アニメーションの設定
    }
}
```

x方向の速度は「そのまま」

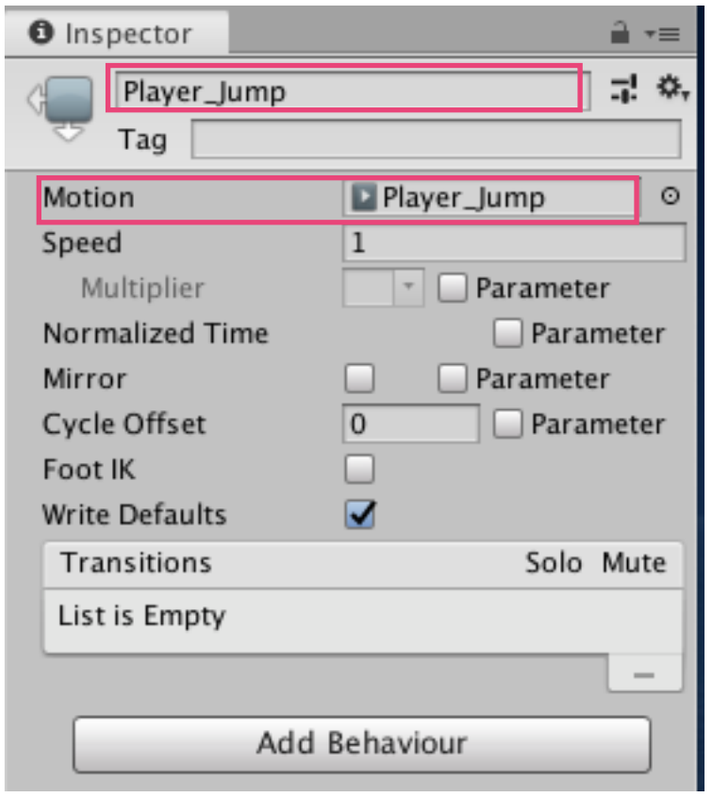
実行してみよう！

ジャンプするアニメーションが欲しい…

Animator Controllerにジャンプを追加しよう



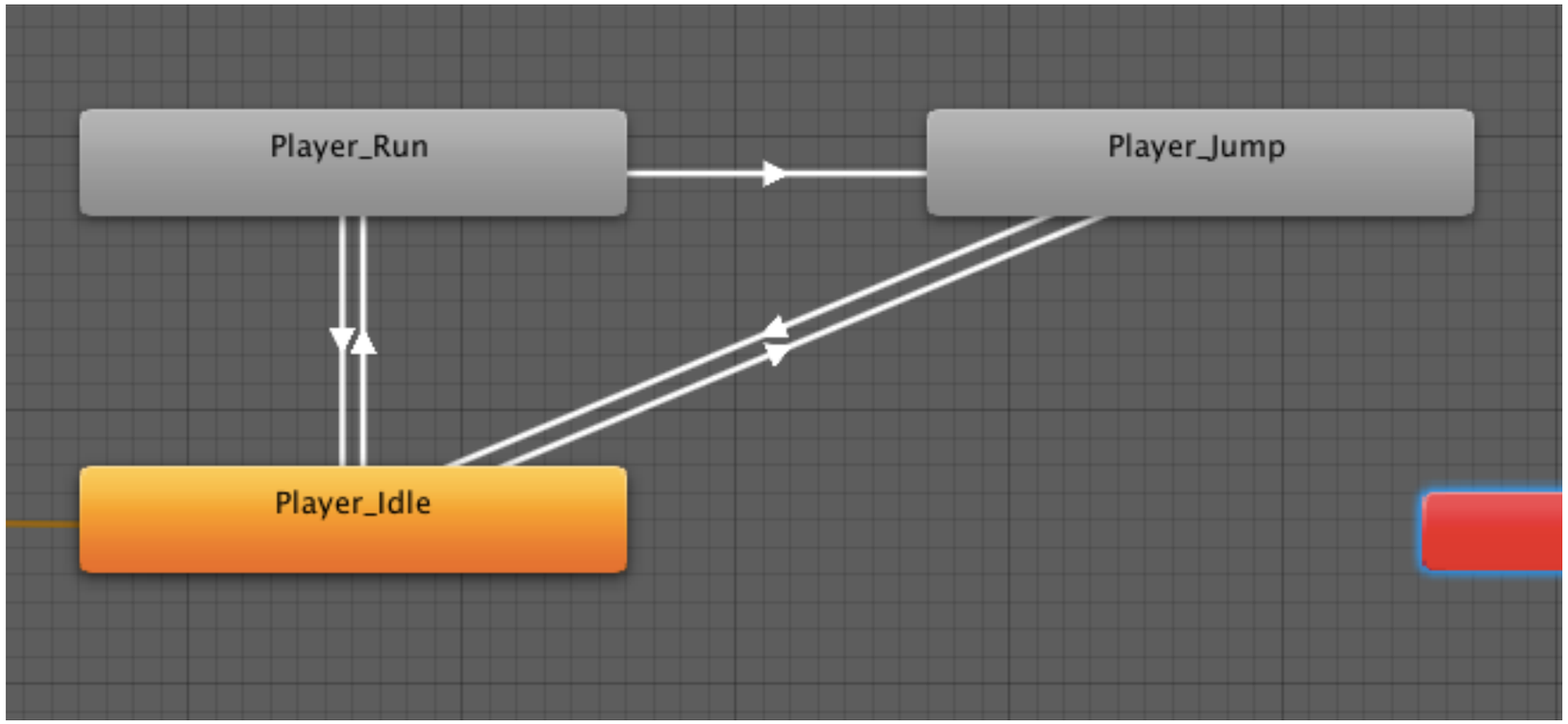
新たなステイトの作成



名前：Player_Jump

Motion：
「Player_Jump」アニメーション

Animator Controllerにジャンプを追加しよう

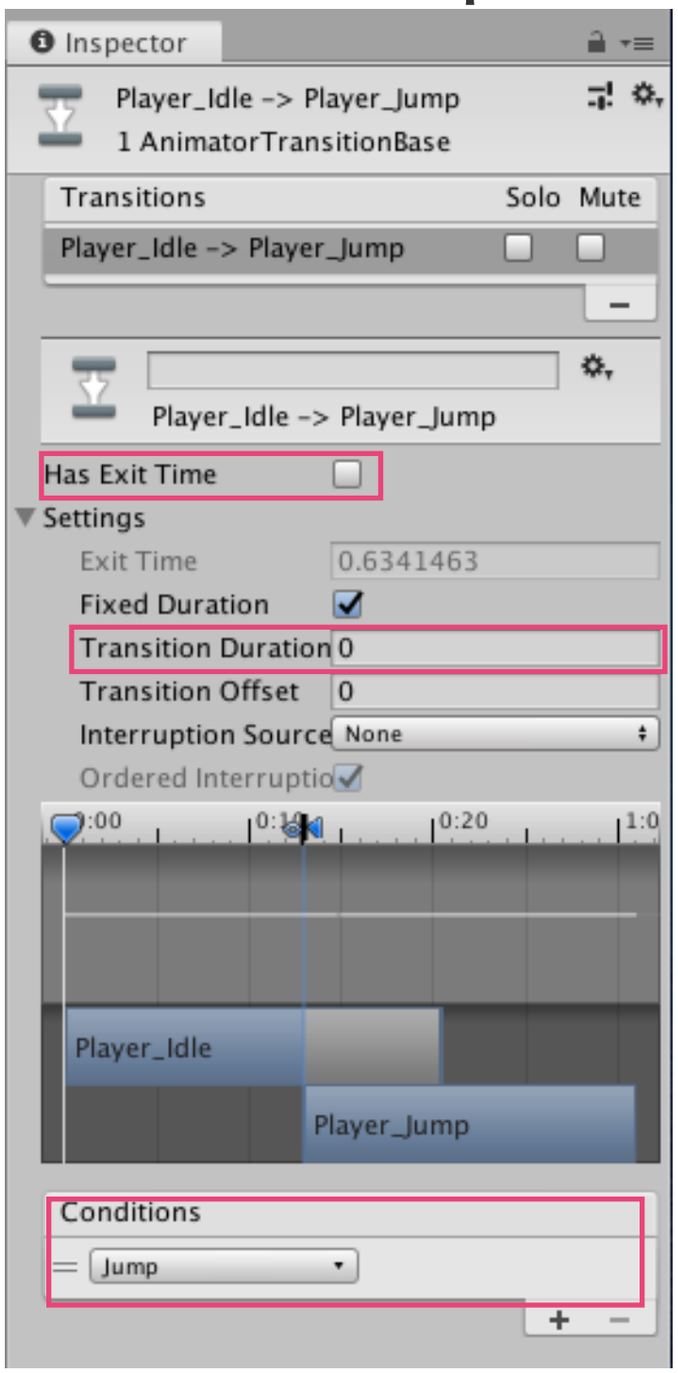


上のように矢印で繋ごう

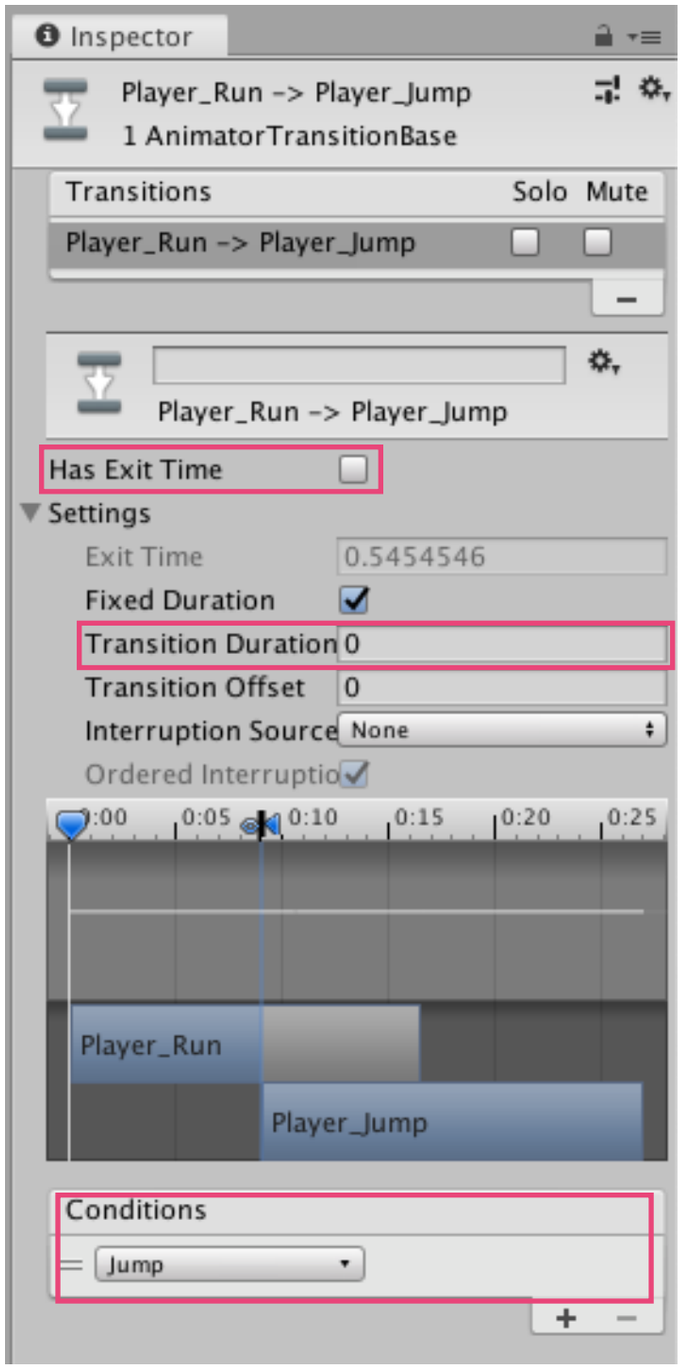
新たな矢印は3つ！

Animator Controllerにジャンプを追加しよう

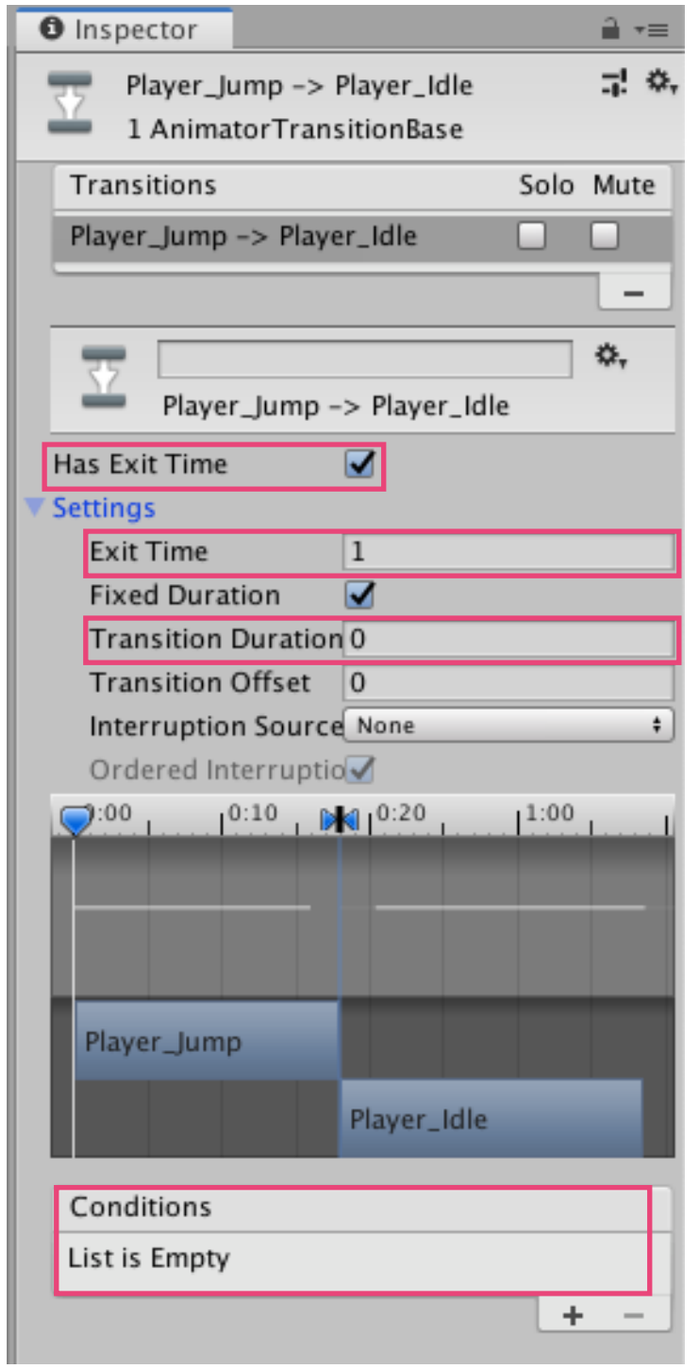
Idle→Jump



Run→Jump



Jump→Idle



↑これだけ少し違うから注意！

問題4-3：Jumpアニメーションに遷移するようにしよう！

・条件

Animatorでのパラメータの名前は「Jump」，型は「Trigger」

```
void Update()
{
```

~中略~

```
    //地面チェック
    GroundCheck();
    if (grounded) //もし地面だったら
    {
        //問題4-1：スペースキー入力
        if(Input.GetKeyDown(KeyCode.Space))
        {
            //問題4-2：速度の設定
            rb2D.velocity = new Vector2(rb2D.velocity.x, 35);

            //問題4-3：アニメーションの設定
                        ？
        }
    }

}
```

□ ←わからない時は、チェックして先に進もう！

注意

次のスライドには答えが書かれています

戻る

もう少し頑張りたい！

進む

できた！ or ギブアップ…

問題4-3：Jumpアニメーションに遷移するようにしよう！

・条件

Animatorでのパラメータの名前は「Jump」，型は「Trigger」

```
void Update()
{
```

~中略~

```
    //地面チェック
    GroundCheck();
    if (grounded) //もし地面だったら
    {
        //問題4-1：スペースキー入力
        if(Input.GetKeyDown(KeyCode.Space))
        {
            //問題4-2：速度の設定
            rb2D.velocity = new Vector2(rb2D.velocity.x, 35);

            //問題4-3：アニメーションの設定
            animator.SetTrigger("Jump");
        }
    }

}
```

「Animator.Set ~ 」でパラメータをセットする！

実行してみよう！

ジャンプアニメーションが再生できた！

まだ宝箱には触れない…

問題4-4：宝箱をタグで判定しよう！
問題4-5：変えたいTextをスクリプトと結びつけよう！
問題4-6：Textの中身を変えよう！

```
//問題3-4：宣言
1 reference
public Text mission1Text;

//問題4-5：宣言
0 references
                ?
```

```
//＝＝＝コード（当たり判定）＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝
//問題3-1
0 references
void OnTriggerEnter2D(Collider2D other)
{
    //問題3-2
    if(other.tag == "Coin")
    {
        //問題3-5
        mission1Text.text = "ミッション 1：CLEAR!";

    }

    //問4-4
    if(                ?                )
    {
        //問4-6
                    ?
    }

}
```

変数名：mission2Text

もしタグが
「Treasure」だったら

mission2Textの中身を
「ミッション 2：CLEAR!」
に変更する

□ ←わからない時は、チェックして先に進もう！

注意

次のスライドには答えが書かれています

戻る

もう少し頑張りたい！

進む

できた！ or ギブアップ…

問題4-4：宝箱をタグで判定しよう！
問題4-5：変えたいTextをスクリプトと結びつけよう！
問題4-6：Textの中身を変えよう！

```
//問題3-4：宣言
1 reference
public Text mission1Text;

//問題4-5：宣言
0 references
public Text mission2Text;
```

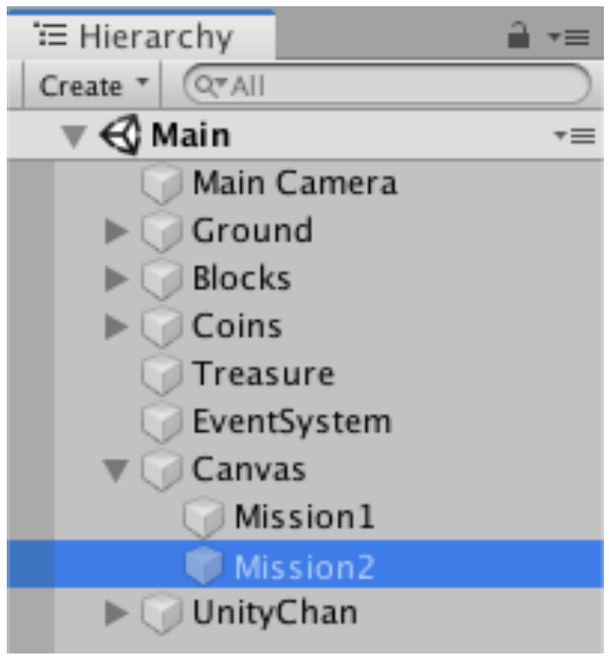
変数名：mission2Text

```
//＝＝＝コード（当たり判定）＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝
//問題3-1
0 references
void OnTriggerEnter2D(Collider2D other)
{
    //問題3-2
    if(other.tag == "Coin")
    {
        //問題3-5
        mission1Text.text = "ミッション 1：CLEAR!";

    }

    //問4-4
    if(other.tag == "Treasure")
    {
        //問4-6
        mission2Text.text = "ミッション 2：CLEAR!";
    }

}
```

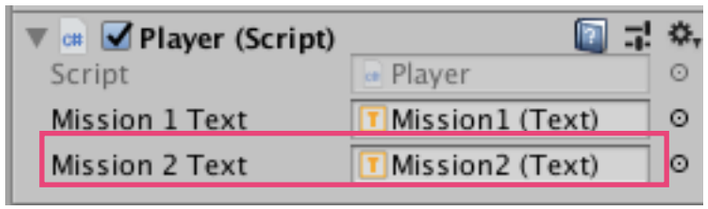
もしタグが
「Treasure」だったら

mission2Textの中身を
「ミッション 2：CLEAR!」
に変更する

変えたいTextをスクリプトと結びつけよう！

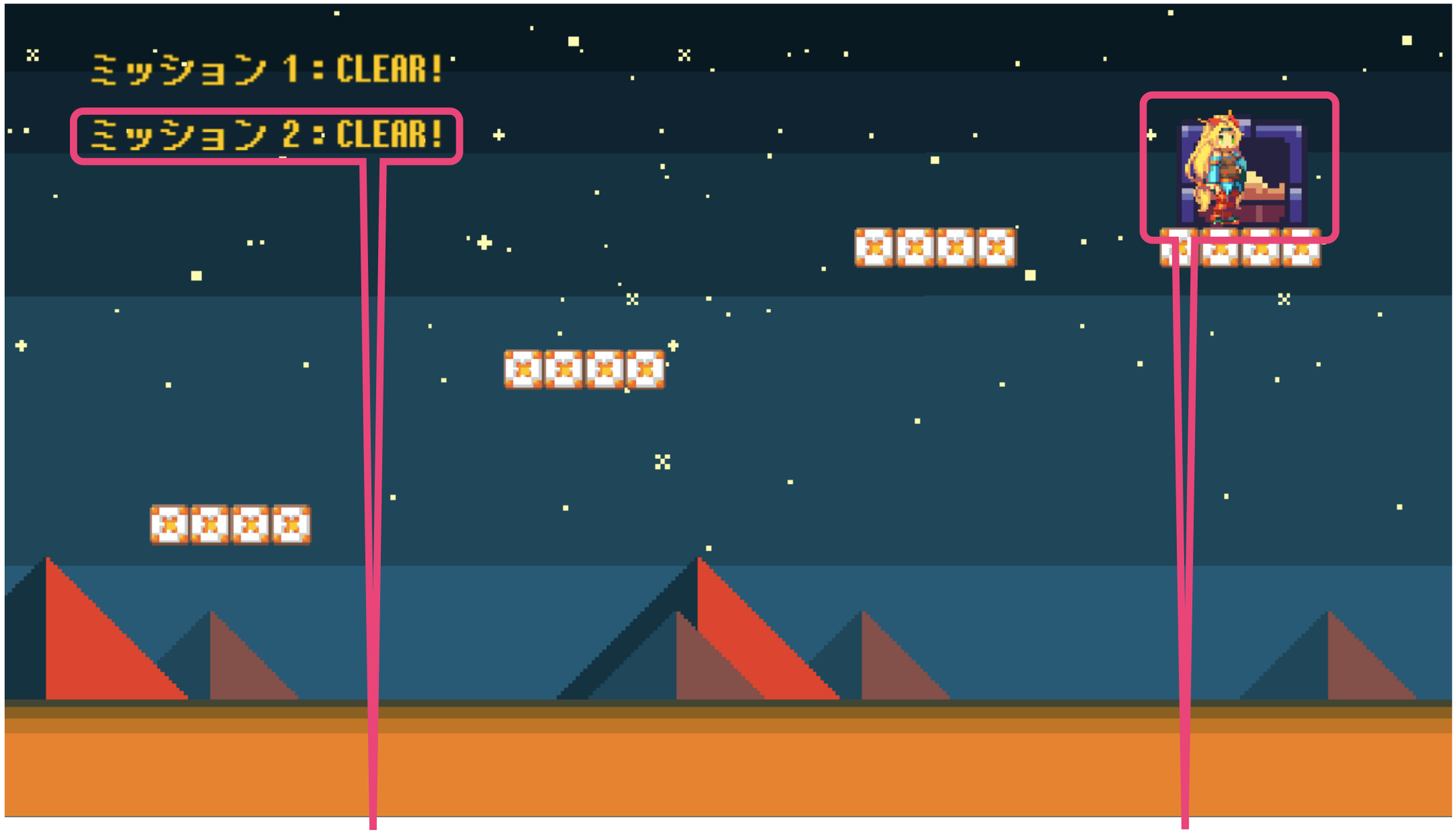


Mission2テキスト



Playerスクリプトに
関連付けしよう！

実行してみよう！



文字が変わる！

宝箱をゲット！

最後に

チェックがついた問題を
復習しよう！

お疲れ様でした！

FIN.